ICONA: Inter Cluster ONOS Network Application

Matteo Gerola[†], Michele Santuari[†], Elio Salvadori[†], Stefano Salsano[§],
Mauro Campanella[‡], Pier Luigi Ventre[‡], Ali Al-Shabibi[*], William Snow[*]

[†]CREATE-NET [§]CNIT / Univ. of Rome Tor Vergata [‡]Consortium GARR [*]Open Networking Laboratory

ABSTRACT

Several Network Operating Systems (NOS) have been proposed in the last few years for Software Defined Networks; however, a few of them are currently offering the resiliency, scalability and high availability required for production environments. Open Networking Operating System (ONOS) is an open source NOS, designed to be reliable and to scale up to thousands of managed devices. It supports multiple concurrent instances (a cluster of controllers) with distributed data stores. A tight requirement of ONOS is that all instances must be close enough to have negligible communication delays, which means they are typically installed within a single datacenter or a LAN network. However in certain wide area network scenarios, this constraint may limit the speed of responsiveness of the controller toward network events like failures or congested links, an important requirement from the point of view of a Service Provider. This paper presents ICONA, a tool developed on top of ONOS and designed in order to extend ONOS capability in network scenarios where there are stringent requirements in term of control plane responsiveness. In particular the paper describes the architecture behind ICONA and provides some initial evaluation obtained on a preliminary version of the tool.

Keywords

Software Defined Networking, Open Source, Geographical Distributed Control Plane, Open Networking Operating System, Distributed Systems, Compute Clusters

1. INTRODUCTION

Since the beginning of the Software Defined Networking (SDN) revolution, the control plane reliability, scalability and availability were among the major concerns expressed by Service and Cloud Providers. Existing deployments show that standard IP/MPLS networks natively offer fast recovery in case of failures. Their main limitation lies in the complexity of the distributed control plane, implemented in the forwarding devices. IP/MPLS networks fall short when it comes to design and implementation of new services that require changes to the distributed control protocols and service logic. The SDN architecture, that splits data and control planes, simplifies the introduction of new services, moving the intelligence from the physical devices to a Network Operating System (NOS), also known as controller, that is in charge of all the forwarding decisions. The NOS is usually considered *logically centralized* since it cannot introduce any single point of failure in production environments. Several *distributed* NOS architectures have been proposed recently to guarantee the proper level of redundancy in the control plane: ONIX [1], Kandoo [2], HyperFlow [3] to name a few.

One of the most promising solutions to properly deal with control plane robustness in SDN is ONOS (Open Networking Operating System) [4]. ONOS offers a stable implementation of a distributed NOS and it has been recently released under a liberal open-source license, supported by a growing community of vendors and operators. In ONOS architecture, a cluster of controllers shares a logically centralized network view: network resources are partitioned and controlled by different ONOS instances in the cluster and resilience to faults is guaranteed by design, with automatic traffic rerouting in case of node or link failure. Despite the distributed architecture, ONOS is designed in order to control the data plane from a single location, even in case of large Wide Area Network scenarios.

However, as envisioned in the work of Heller et al. [5], even though a single controller may suffice to guarantee round-trip latencies on the scale of typical mesh restoration target delays (200 msec), this may not be valid for all possible network topologies. Furthermore, ensuring an adequate level of fault tolerance can be guaranteed only if controllers are placed in different locations of the network.

To this purpose, we designed a geographically distributed multi-cluster solution called ICONA (Inter Cluster ONOS Network Application) which is working on top of ONOS and whose purpose is to both increase the robustness to network faults by redounding ONOS clusters in several locations but also to decrease event-to-response delays in large scale networks. In the rest of the paper we will consider a large scale Wide Area Network use-case under the same administrative domain
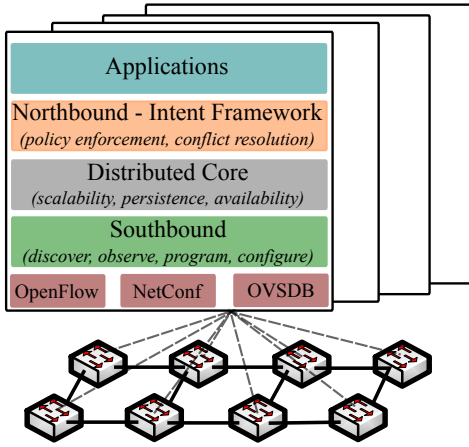
1

Figure 1: ONOS distributed architecture

(e.g. single service provider), managed by a single logical control plane. ICONA is a tool that can be deployed on top of ONOS; a first release is available under the Apache 2.0 open source license [6].

The structure of the paper is as follows. Sect. 2 provides an overview of ONOS, while Sect. 3 presents the ICONA architecture. Some preliminary results are then discussed in Sect. 4. Sect. 5 discusses the state-of-the-art in the field. Finally, Sect. 6 draws conclusions and indicates future works.

2. AN OVERVIEW OF ONOS

ONOS (Open Network Operating System) is a distributed SDN controller that has been recently released as open source by ON.Lab as part of a joint community effort with a number of partners including AT&T, NEC and Ericsson among the others [7].

ONOS implements a distributed architecture in which multiple controller instances share multiple distributed data stores with eventual consistency. The entire data plane is managed simultaneously by the whole cluster. However, for each device a single controller acts as a master, while the others are ready to step in if a failure occurs. With these mechanisms in place, ONOS achieves scalability and resiliency. Figure 1 shows the ONOS internal architecture within a cluster of four instances. The Southbound modules manage the physical topology, react to network events and program/configure the devices leveraging on different protocols. The Distributed Core is responsible to maintain the distributed datastores, to elect the master controller for each network portion and to share information with the adjacent layers. The NorthBound modules offer an abstraction of the network and the interface for application to interact and program the NOS. Finally, the Application layer offers a container in which third-party applications can be deployed.

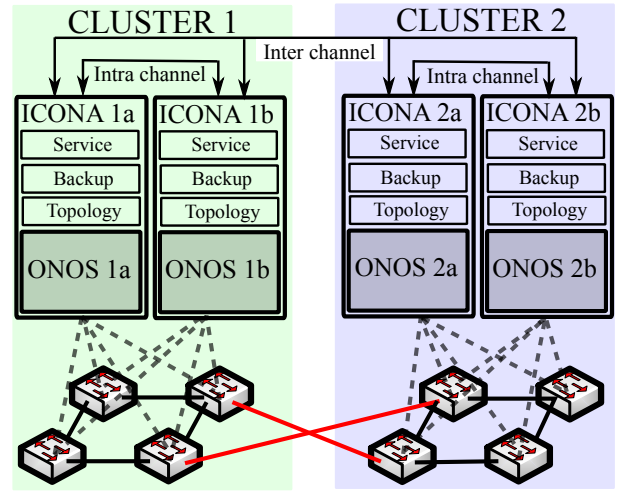In case of a failure in the data plane (switch, link or

Figure 2: ICONA architecture: the data plane is divided in portions interconnected through so-called inter-cluster links (in red). Each portion is managed by multiple co-located ONOS and ICONA instances, while different clusters are deployed in various data centers.

port down), an ONOS instance detects the event (using the Southbound modules), computes alternative paths for all the traffic crossing the failed element (operation in charge of the Intent Framework), and publishes them on the Distributed Core: then, each master controller configures accordingly its network portion.

ONOS leverages on Apache Karaf [8], a java OSGi based runtime, which provides a container onto which various component can be deployed. Under Karaf, an application, such as ICONA, can be installed, upgraded, started and stopped at runtime, without interfering other components.

3. ICONA ARCHITECTURE

The basic idea behind ICONA is to partition the Service Provider's network into several geographical regions, each one managed by a different cluster of ONOS instances. While the management of the network is still under the same administrative domain, a peer-to-peer approach allows a more flexible design. The network architect can select the number of clusters and their geographical dimension depending on requirements (e.g. leveraging on some of the tools being suggested within the aforementioned work [5]), without loosing any feature offered by ONOS, neither worsening the system performances. In this scenario, each ONOS cluster provides both scalability and resiliency to a small geographical region, while several clusters use a publish-subscribe event manager to share topology information, monitor events and operator's requests.

The ICONA control plane is geographically distributed among different data centers, each one controlling

the adjacent portion of the physical network. Figure 2 shows the aforementioned architecture where each cluster is located in a different data center (e.g. ONOS 1a and ONOS 1b are co-located instances). To offer the same reliability already available in ONOS, the ICONA application runs on top of each instance. Referring on Figure 1, ICONA is deployed in the Application layer, and interacts with ONOS using the Northbound - Intent Framework. Inside each cluster, a master controller is elected, using a distributed registry. The ICONA master is in charge of sharing information and applying decisions, while all the backups are aware of the network status, and can become master in case of failure.

The communication between different ICONA instances, both intra and inter-cluster, is currently based on Hazelcast [9]. Hazelcat offers a multicast event-based messaging system, inspired by Java Messaging System (JMS). Applications (e.g. controllers) can publish a message onto a topic, which will be distributed to all instances of the application that have subscribed to that topic. Hazelcast does not have a single point of failure that is not achieved easily by pure JMS solutions. In ICONA, one Hazelcast channel is devoted to the intra-cluster communication (e.g. local ONOS cluster) and another one for the inter-cluster messages. ICONA runs as bundle in the ONOS Karaf container, taking advantages of all the features mentioned in Sect. 2.

In devising a geographical architecture, covering thousands of square kilometers, a key element is the type and amount of information that the different segments of the control plane have to share. Control traffic has to be minimized and the system has to optimize is performance in terms of:

- offering an up-to-date view of the network, including status of the nodes,

- configuring the network devices,

- reacting to failures both in data and control plane without disrupting customer's traffic.

Three internal modules implement these features in ICONA (see Figure 2): the Topology Manager, the Service Manager and the Backup Manager. In the following Sections an overview of each of these components is provided.

3.1 Topology Manager

The Topology Manager (TM) is responsible to discover the data plane elements, reacts to network events, stores in a persistent local database the links and devices with the relevant metrics. To offer a partial view of its network portion, each TM shares with the other ICONA clusters the following information through the inter-cluster channel:

- Inter-cluster link (IL): an IL is the physical connection between two clusters. ICONA implements an enhanced version of the ONOS discovery mechanism, based on the Link Layer Discovery Protocol (LLDP). Each IL is shared with all the clusters tagged by some metrics, such as the link delay, available bandwidth and number of flows crossing the link.

- End-point (EP) list: an EP defines the interconnection between the customer's gateway router and the ICONA network. Each cluster shares the list of its EPs and the metrics (bandwidth, delay) between these EPs and all the clusters ILs.

3.2 Service Manager

All the interaction between the network operators and the control plane are handled by the Service Manager (SM). This module offers a REST API used to poll ICONA for network events and alarms, and to manage (instantiate, modify and delete) the services. In our initial implementation, we have considered only two services: L2 pseudo-wire tunnels and MPLS VPN overlay networks, which are key services in a Service provider network. However, the application can be easily extended to provide other functionalities.

The implemented services require interconnecting two or multiple EPs that can be controlled by the same ICONA cluster or by different ones. The SM computes the overall metrics and chooses the best path between two EPs. There are two cases:

- If they belong to the same cluster, it directly asks ONOS to install an OpenFlow 1.3 [10] MPLS-based flow path in the physical devices.

- If they belong to two different clusters, the SM follows this procedure:

  - It contacts all the involved clusters (e.g. the ones that are crossed by this service request) asking to reserve the local portion of the path.

  - As soon as all the clusters have replied to the reservation event (if at least one does has a negative reply, the SM releases the path and computes an alternative ones), it requests to switch from the reservation status to the installation. Each ICONA cluster master then contacts the local ONOS asking to install the flow path in the physical devices.

  - If the second step is successful too, the SM ends the procedure, otherwise it asks all the clusters to release the resources, computes an alternative route and restarts the procedure.

Finally, the SM updates all the ICONA clusters with the new installed service and the relevant information.

While this path installation procedure is slower than the original implemented by ONOS (around three times), ICONA cannot assume a strong consistency between the remote databases. In the aforementioned ISP scenarios, the amount of time required for a service installation is not a relevant metric, but the procedure has to be robust enough to tolerate substantial delays.

3.3 Backup Manager

If a failure happens within the same cluster, ONOS takes care of rerouting all the paths involved in the failure. After receiving a `PORT_STATUS` or `LINK_DOWN` OpenFlow message, ONOS detects all the flows crossing the failed element and computes an alternative path to the destination. Finally, it installs the new flows in the network and removes the old ones.

If instead the failure happens between two different clusters (e.g. it involves an IL or one of the two nodes that sits at the edge of the IL) then the ICONA's module named Backup Manager (BM) will handle the failure. In particular:

- For each IL, a backup link (BIL) completely decoupled from the primary one, is computed and pre-installed in the data plane. The BIL is a virtual link selected among all the available paths between the source and destination ICONA clusters, taking into account composite metrics, such as the delay, available bandwidth and numbers of flows.

- When the failure happens, all the traffic crossing the IL is rerouted to the BIL by each ICONA edge cluster, without the need to wait for remote instances to share any information.

- When the rerouting is completed, the new path is notified to all the remote clusters, in order to share the same network status.

The amount of time required for the rerouting is particularly relevant in an ISP network because it is most directly related to SLAs guaranteed by network operators.

4. EVALUATION

The purpose of the experimental tests described in this section is to compare ICONA with a standard ONOS setup, and evaluate the performances of the two solutions in an emulated environment.

The control plane is always composed of 8 virtual machines, each with four Intel Core i7-2600 CPUs @ 3.40GHz and 8GB of RAM. While for the ONOS tests all the instances belong to the same cluster, with ICONA we have created 2, 4 and 8 clusters, respectively with 4, 2 and 1 instances each. The data plane is emulated by Mininet [11] and Netem [12]: the former creates and
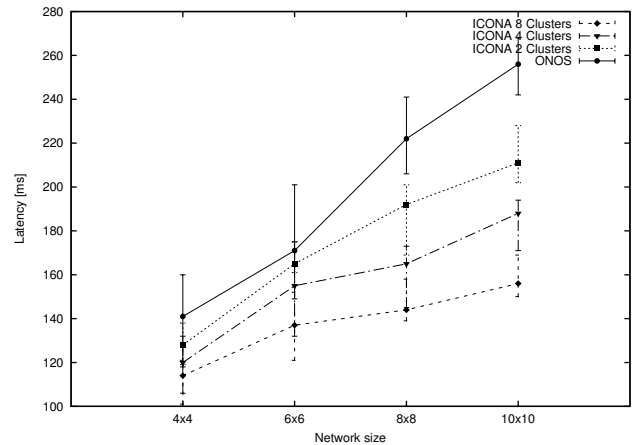

Figure 3: Average, maximum and minimum latency to reroute 100 paths in case of link failure for ONOS and ICONA (2, 4 and 8 clusters)

manages the OpenFlow 1.3 network, while the latter emulates the properties of wide area networks, such as variable delay, throughput and packet loss. Both solutions (ONOS and ICONA) have been tested with both a regular (grid) topology and with the topology of the GÉANT [13] pan-European network. It is important to highlight that the current ONOS release (Avocet 1.0.1) is focusing on functions and correctness, while Blackbird, to be released in March, will dramatically improve the internal performance. For these reasons, the results presented in this section should not be considered as benchmark.

4.1 Reaction to Network Events

4.1.1 Grid networks

The first performance metric is the overall latency of the system for updating the network state in response to events; examples include rerouting traffic in response to link failure or moving traffic in response to congestion. To evaluate how the system performs when the forwarding plane scales-out, we have selected some standard grid topologies, with a fixed link delay of 5ms (one-way) and then we have been comparing the latency needed to reroute a certain number of installed paths when an inter-cluster link fails for both ONOS and ICONA with various clustering settings.

We define as overall latency the amount of time that both ONOS and ICONA require to react to the failure as the sum of: i) the amount of time for the OpenFlow messages (`PORT_STATUS` and `FLOW_MOD`) to traverse the control network, ii) the alternative path computation, iii) the installation of new flows in the network devices and iv) the deletion of the pre-existing flows. In particular, we have been running several simulations by installing one thousand paths in the network and then
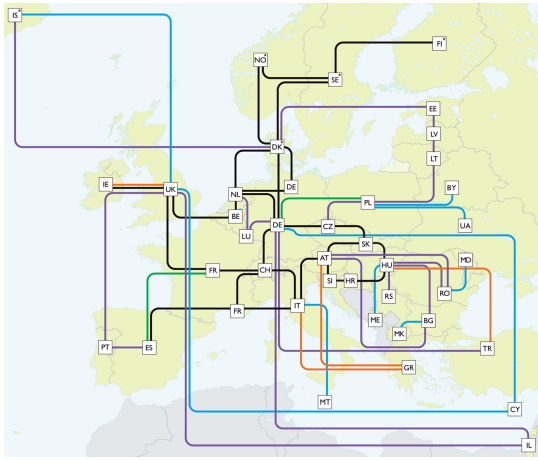
Figure 4: GÉANT pan-European network

by making fail an inter-cluster link with al least one hundred pseudo-wires running.

Figure 3 shows the latency (avg, min, max) required for the different solutions to execute the four tasks previously mentioned. Each test has been repeated 20 times. Despite the same mechanism used by ICONA to compute and install the new paths, the difference is mainly due to the following reasons: i) each ICONA cluster is closer to the devices, thus reducing the amount of time required for OpenFlow messages to cross the control channel and ii) the ICONA clusters are smaller, with fewer links and devices, thus decreasing the computation time and the overall numbers of flows to be installed and removed from the data plane.

4.1.2 GEANT network

The same metrics have been evaluated on the GÉANT topology (see Figure 4). Circuits have various one-way delays (from 10 to 50ms) and throughputs (from 1 to 100Gbps).

Control plane	Avg latency [ms]	Min latency [ms]	Max latency [ms]
ONOS	297	284	308
ICONA2	272	261	296
ICONA4	246	232	257
ICONA8	221	199	243

Table 1: GÉANT network: average, maximum and minimum latency to reroute 100 paths in case of link failure for ONOS and ICONA (2, 4 and 8 clusters)

Table 1 depicts similar results as the previous test. ONOS has been compared with three different ICONA deployments, with 2, 4 and 8 clusters, respectively with 4, 2 and 1 instances each. While having the same num-

ber of VMs running the control plane software, their geographical distribution improves the overall performance of the system. While the network is smaller than the 10*10 grid topology, with 41 switches and 58 bi-directional links, the higher delay in the data plane requires an additional amount of time to reconverge.

4.2 Startup Convergence Interval

This second experiment measures the overall amount of time required for both solutions to re-converge after a complete disconnection between the control and data planes. The tests have been performed over the GÉANT topology, and replicated 20 times. Table 2 shows the average, maximum and minimum values in seconds.

Control plane	Average Time [s]	Minimum Time [s]	Maximum Time [s]
ONOS	6,98	6,95	7,06
ICONA	6,96	6,88	7,02

Table 2: Amount of time required to obtain the network convergence after disconnection for ONOS and ICONA

The result shows that ICONA and ONOS require comparable time intervals to return to a stable state, in case of a complete shutdown or a failure of the control plane.

5. RELATED WORK

The logical centralization of the control plane advocated by the SDN approach requires a specific design in terms of control network performance, scalability and fault tolerance. Most of the open source controllers currently available are focused on functionalities more than on scalability and fault tolerance. This section provides a review about distributed architectures for the SDN control plane, that address the scalability and fault tolerance issues. ONIX [1] provides an environment on top of which a distributed NOS can be implemented with a logically centralized view. The distributed Network Information Base (NIB) stores the state of network in the form of a graph; the platform is responsible for managing the replication and distribution of the NIB, the applications have to detect and resolve conflicts of network state. Scalability is provided through network partitioning and aggregation. Regarding the fault tolerance, the platform provides the basic functions, while the control logic implemented on top of ONIX needs to handle the failures. ONIX has been used in the B4 network, the private WAN [14] that inter-connects Google's data centers around the world. This high level design is similar in our ICONA solution, but ICONA is not tailored to a specific use case, providing a reusable framework on top of which it is possible to build specific applications. The

Kandoo [2] architecture addresses the scalability issue by creating an architecture with multiple controllers: the so-called root controller is logically centralized and maintains the global network state; the bottom layer is composed of local controllers in charge of managing a restricted number of switches. The Kandoo architecture does not focus on the distribution/replication of the root controller and on fault tolerance neither in the data plane nor in the control plane. HyperFlow [3] focuses on both scalability and fault tolerance. Each HyperFlow instance manages a group of devices without losing the centralized network view. A control plane failure is managed by redirecting the switches to another HyperFlow instance. The applicability of such approach to WAN scenarios with large delays between the different HyperFlow instances is not considered. DISCO [15] architecture considers a multi-domain environment. This approach is specifically designed to control a WAN environment, composed of different geographical controllers that exchange summary information about the local network topology and events. This solution overcomes the HyperFlow limitations, however it does not provide local redundancy: in the case of a controller failure, a remote instance takes control of the switches, increasing the latency between the devices and their primary controller. ElastiCon [16] and Pratyaastha [17] aim to provide an elastic and efficient distributed SDN control plane to address the load imbalances due to static mapping between switches and controllers and spatial/temporal variations in the traffic patterns. SMaRtLight [18] considers a distributed SDN controller aiming at a fault-tolerant control plane. It only focuses on control plane failures, assuming that data plane failures are dealt with by SDN applications on top of the control platform. The Beacon cluster [19] project also targets the datacenter scenario, with a general framework focusing on load balancing (with addition and removal of controllers), dynamic switch-to-controller mapping and instance coordination. In OpenDaylight [20], an initial work on clustering has been provided in the last release of the project (Helium) using the Akka framework [21] and the RAFT consensus algorithm [22].

6. CONCLUSIONS

In this paper we have presented ICONA (Inter Cluster ONOS Network Application), a tool built on top of ONOS distributed controller [4] whose aim is to enable clustering of ONOS instances across different locations on a Wide Area Network. In fact, while current ONOS release may scale and perform well on a variety of network topologies, it may not suffice for all, especially when there are latency bounds in restoration scenarios. ICONA is based on a pragmatic approach that inherits ONOS features while improving its responsiveness to

network events like link/node failures or congested links that impose the rerouting of large set of traffic flows.

A preliminary release of ICONA is available under permissive open source license and we are collaborating with ON.Lab to evaluate the opportunity to include some of the ideas developed therein as part of the official release of ONOS. Future improvements of ICONA will focus on solutions to improve the path installation procedure and on combining clustering design techniques with ICONA deployment in order to guarantee the best tradeoff between performances and replication of clusters in different network locations.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for large-scale production networks. In *9th USENIX Conference on Operating Systems Design and Implementation*, 2010.

[2] H. Y. Soheil and Y.Ganjali. Kandoo: a framework for efficient and scalable offloading of control applications. In *First workshop on Hot topics in in software defined networking*, 2012.

[3] A. Tootoonchian and Y. Ganjali. Hyperflow: a distributed control plane for openflow. In *2010 internet network management conference on Research on enterprise networking*, 2010.

[4] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar. Onos: towards an open, distributed sdn os. In *Third workshop on Hot topics in in software defined networking*, 2014.

[5] B. Heller, R. Sherwood, and N. McKeown. The controller placement problem. In *First workshop on Hot topics in in software defined networking*, 2012.

[6] Icona software - https://github.com/smartinfrastructures/dreamer.

[7] Onos website - http://onosproject.org/.

[8] Apache karaf - http://karaf.apache.org/.

[9] Hazelcast - http://www.hazelcast.com.

[10] Openflow switch specification version 1.5.0 - https://www.opennetworking.org/images/stories/

downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf.

[11] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *9th ACM Workshop on Hot Topics in Networks*, 2010.

[12] Netem - http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[13] Geant - the core national research and education networks (nrens) european backbone - http://www.geant.net/pages/default.aspx.

[14] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *ACM SIGCOMM Computer Communication Review*, 2013.

[15] K. Phemius, M. Bouet, and J. Leguay. Disco: Distributed sdn controllers in a multi-domain environment. In *IEEE Network Operations and Management Symposium*, 2014.

[16] A.A. Dixit, F. Hao, S. Mukherjee, T.V. Lakshman, and R. Kompella. Elasticon: an elastic distributed sdn controller. In *Tenth ACM/IEEE symposium on Architectures for networking and communications systems*, 2014.

[17] A Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson. Pratyaastha: an efficient elastic distributed sdn control plane. In *Third workshop on Hot topics in in software defined networking*, 2014.

[18] F Botelho, A Bessani, F Ramos, and P Ferreira. *SMaRtLight: A Practical Fault-Tolerant SDN Controller*. ArXiv e-prints, 2014.

[19] V Yazici, M. O. Sunay, and A. O. Ercan. Controlling a software-defined network via distributed controllers. In *2012 NEM Summit, pp. 16-20*, 2012.

[20] Opendaylight - http://www.opendaylight.org/.

[21] Akka framework - http://akka.io/.

[22] Raft consensus algorithm - https://raftconsensus.github.io/.